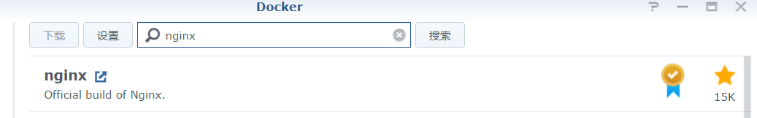
Django +uwsgi+nginx部署流程

1. 建立nginx docker



配置文件为



Note： 端口设置中本地端口为nas端口，容器端口为docker内部nginx端口，此处80必须与nginx.conf配置中一致，否则无法监听服务。



图表 1 nginx conf



1. 在nginx容器中配置python和git环境
   1. 打开docker终端机，新增bash窗口
   2. apt-get update更新source源
   3. apt-get install python3-pip 安装python最新版本，这一步巨慢，并且找不到source.list文件，无法更换国内源
   4. apt-get install python3-venv 安装python虚拟环境包，否则直接pip install会报错PEP 668 “Externally managed environments”
   5. 进入虚拟环境，安装pip包
      1. cd /data
      2. python3 -m venv cat\_venv
      3. source cat\_venv/bin/activate
      4. pip3 install uwsgi
      5. pip3 install Django==4.2.10 (based on requirements.txt)
2. 配置uwsgi
   1. 启动uwsgi服务命令：uwsgi –ini uwsgi.ini
   2. 后台启动命令：nohup uwsgi –ini uwsgi.ini &
   3. 关闭uwsgi命令： pkill -f uwsgi -9
   4. 解读uwsgi.ini

[uwsgi]

socket=127.0.0.1:8000

#http=127.0.0.1:8084

chdir=/data

wsgi-file=catSystem/wsgi.py

static-map=/static=/data/static

master=True

max-requests=5000

processes = 4

threads = 2

vacuum=True

home=/data/cat\_venv

buffer-size=32768

#disable-logging = true

#daemonize=/var/log/uwsgi.log

Socket：用于和nginx服务器进行通讯，端口与nginx.conf中一致

Http: 和socket二选一，用于uwsgi单独部署django服务，此处端口就是对外访问端口

Chdir: 存放django项目目录

Wsgi-file：wsgi文件相对路径

Static-map：静态文件绝对路径

Home：python虚拟环境绝对路径

Daemonize：可能可以直接后台运行uwsgi，代替nohup和&，未尝试

1. 配置nginx
   1. #查看版本号 nginx -v
   2. #启动nginx service nginx start
   3. #重启 service nginx restart
   4. #重载 nginx -s reload
   5. #停止nginx nginx -s stop #直接停止
   6. #完成已经接收的连接请求后再退出 nginx -s quit
   7. #查看nginx进程 ps -ef | grep nginx
   8. 编辑配置文件： /etc/nginx/nginx.conf，参考红字注释

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

# tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 4096;

# include /etc/nginx/conf.d/\*.conf;

server {

listen 80; #对应nginx容器端口

server\_name localhost;

charset utf-8;

client\_max\_body\_size 75M;

# root /root/projectpy/Django根目录;

location / {

include /etc/nginx/uwsgi\_params;

uwsgi\_pass 0.0.0.0:8000; #对应uwsgi socket端口

uwsgi\_read\_timeout 1000;

}

location /static/ {

root /data/; #对应静态目录绝对路径，由黄色两部分拼接

# alias /data/static; #非拼接

}

error\_page 404 /404.html;

location = /404.html {

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

}

#add\_header X-Frame-Options SAMEORIGIN;

#add\_header 'Referrer-Policy' 'same-origin' always;

#add\_header X-Download-Options "noopen" always;

#add\_header X-Permitted-Cross-Domain-Policies "master-only" always;

# add\_header X-XSS-Protection "1; mode=block";

# add\_header X-Content-Type-Options nosniff;

# add\_header Content-Security-Policy "default-src 'self' ";

}

}

1. Django
   1. 常用命令
      1. Python3 manage.py shell 打开django命令行，一般用于手动增删改查数据库条目
      2. Python3 manage.py collectstatic 把项目上的静态文件统一收集到/static下，部署可能会用
      3. Python3 manage.py makemigrations 根据目前models改动生成migrations文件
      4. Python3 manage.py migrate 对比数据库migration table和本地migrations文件，如果不匹配更新数据库（理论上本地永远应该比数据库领先）
      5. python -Wa manage.py test 处理过期警告，升级django版本时用
      6. Python3 manage.py runserver 最常用命令，用于开发环境或查看服务端是否有编译错误